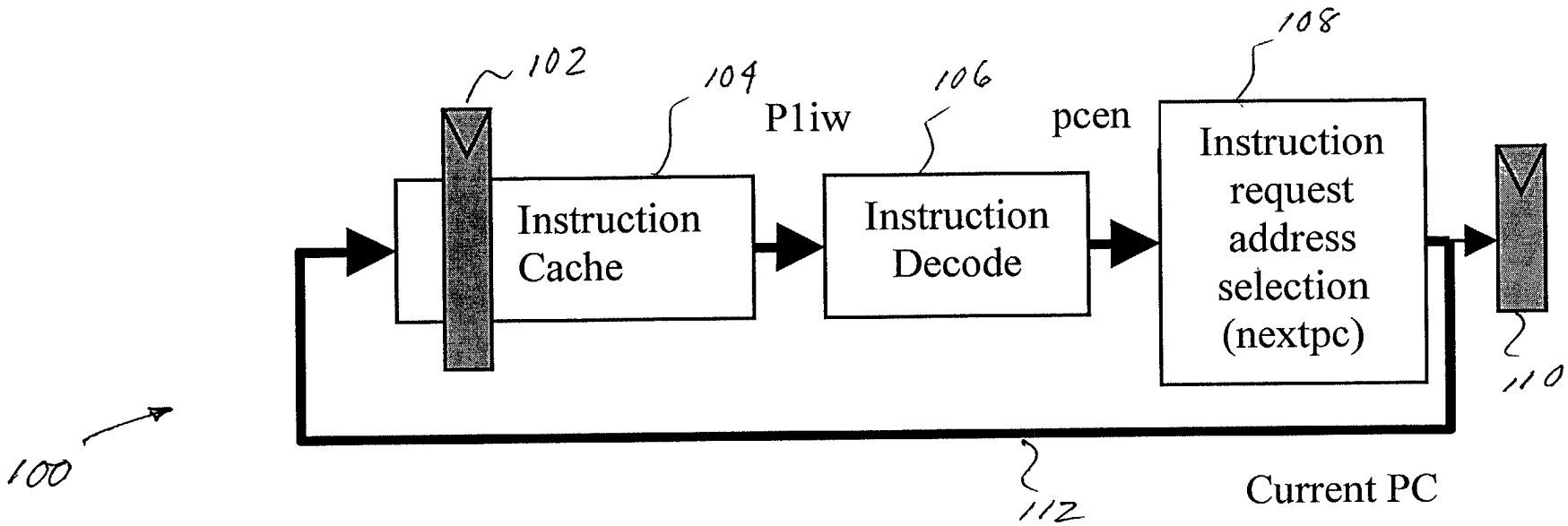

102
104
P1iw
106
pcen
108
Instruction Cache
Instruction Decode
Instruction request address selection (nextpc)
110
100
112
Current PC

Fig. 1

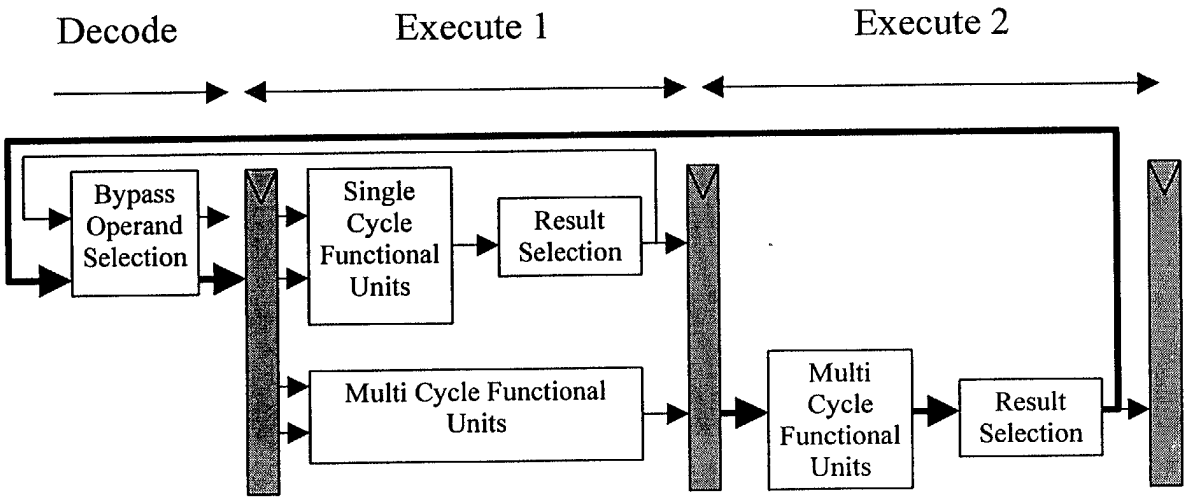


Fig. 2

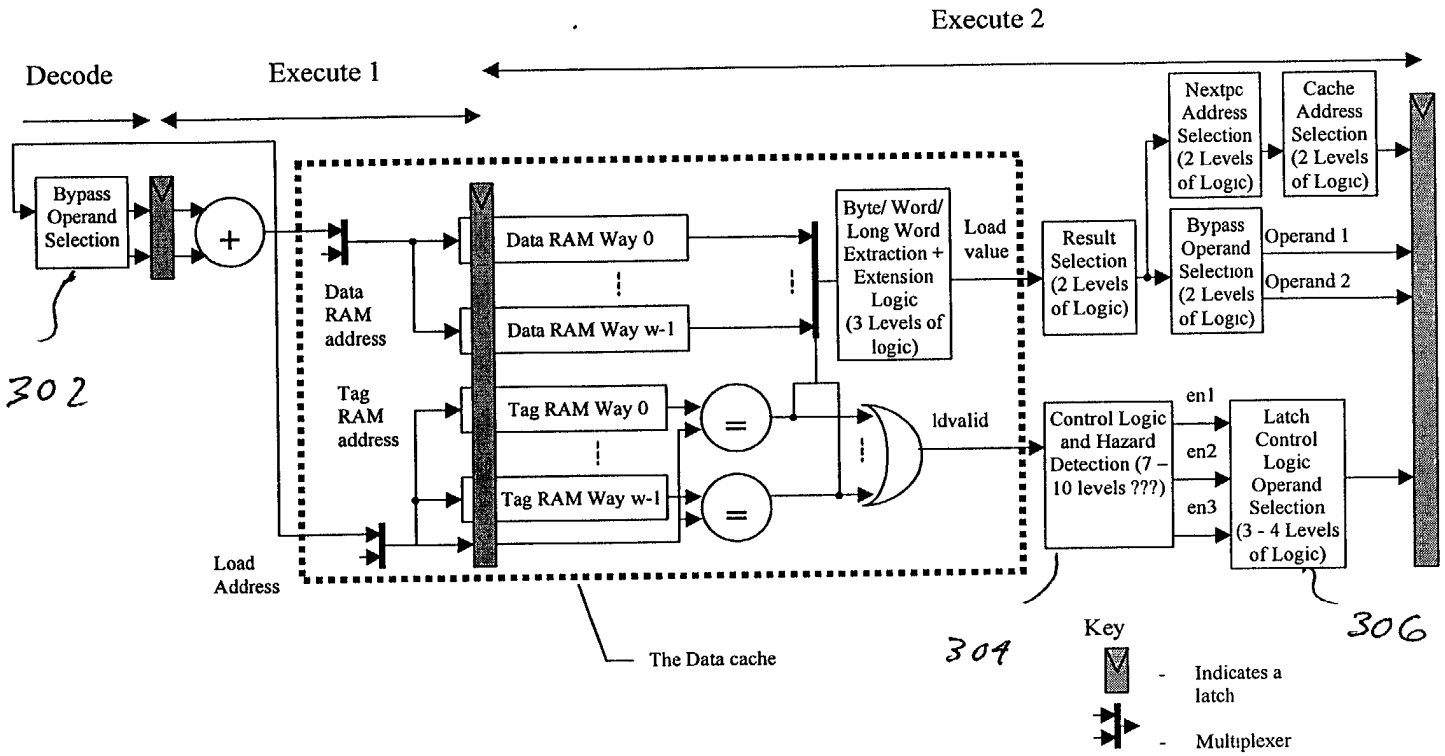

Execute 2
Decode
Execute 1
Bypass Operand Selection
+
Data RAM address
Tag RAM address
Load Address
Data RAM Way 0
Data RAM Way w-1
Tag RAM Way 0
Tag RAM Way w-1
=
=
Byte/ Word/ Long Word Extraction + Extension Logic (3 Levels of logic)
Load value
Idvalid
Result Selection (2 Levels of Logic)
Nextpc Address Selection (2 Levels of Logic)
Cache Address Selection (2 Levels of Logic)
Bypass Operand Selection (2 Levels of Logic)
Operand 1
Operand 2
Control Logic and Hazard Detection (7 – 10 levels ???)
en1
en2
en3
Latch Control Logic Operand Selection (3 - 4 Levels of Logic)
302
304
306
The Data cache
Key
- Indicates a latch
- Multiplexer

Fig. 3

Fig. 3a.

	Pipeline Stage					
Reference Step	0	F	D	E1	E2	WB
350	Ld					
352	Mov	Ld				
354	Add	Mov	Ld			
356		Add	Mov	Ld		
358			Add	Mov	Ld	
360			Add		Mov/Ld	
362			Add		Ld	Mov
364				Add		Ld
366					Add	
368						Add

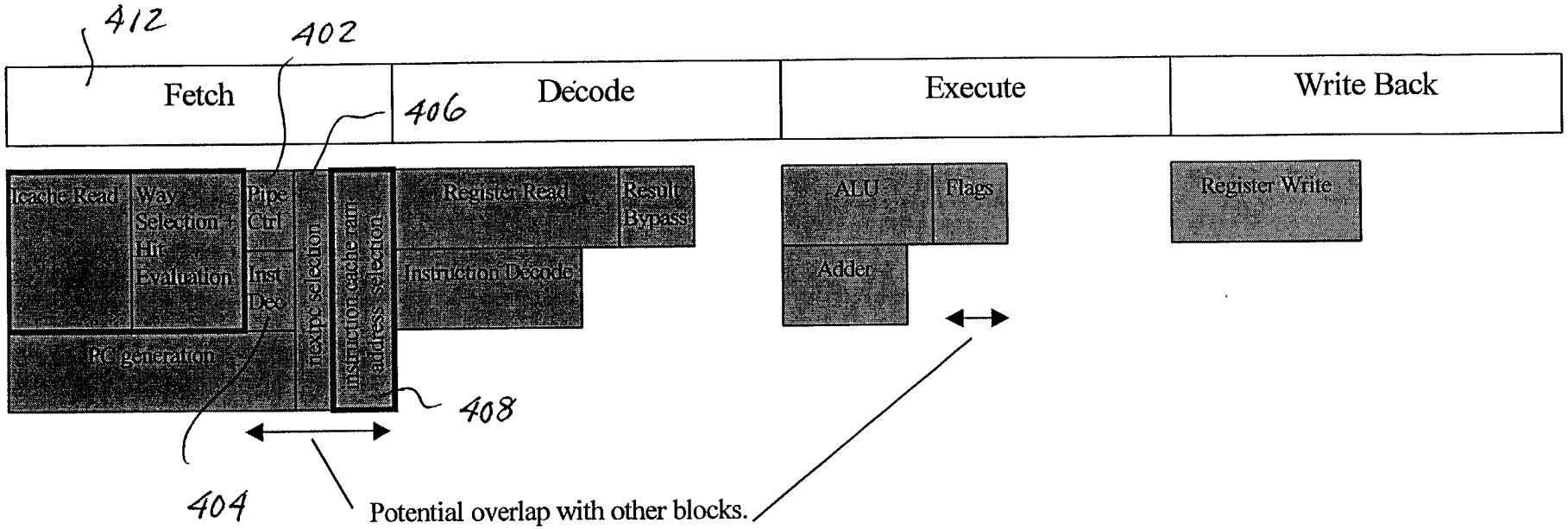

412
402
406
Fetch
Decode
Execute
Write Back
Icache Read
Way Selection + Hit Evaluation
Pipe Ctrl
Inst Dec
PC generation
nextpc selection
instruction cache ram address selection
Register Read
Result Bypass
Instruction Decode
ALU
Flags
Adder
Register Write
408
404
Potential overlap with other blocks.

Fig. 4

400

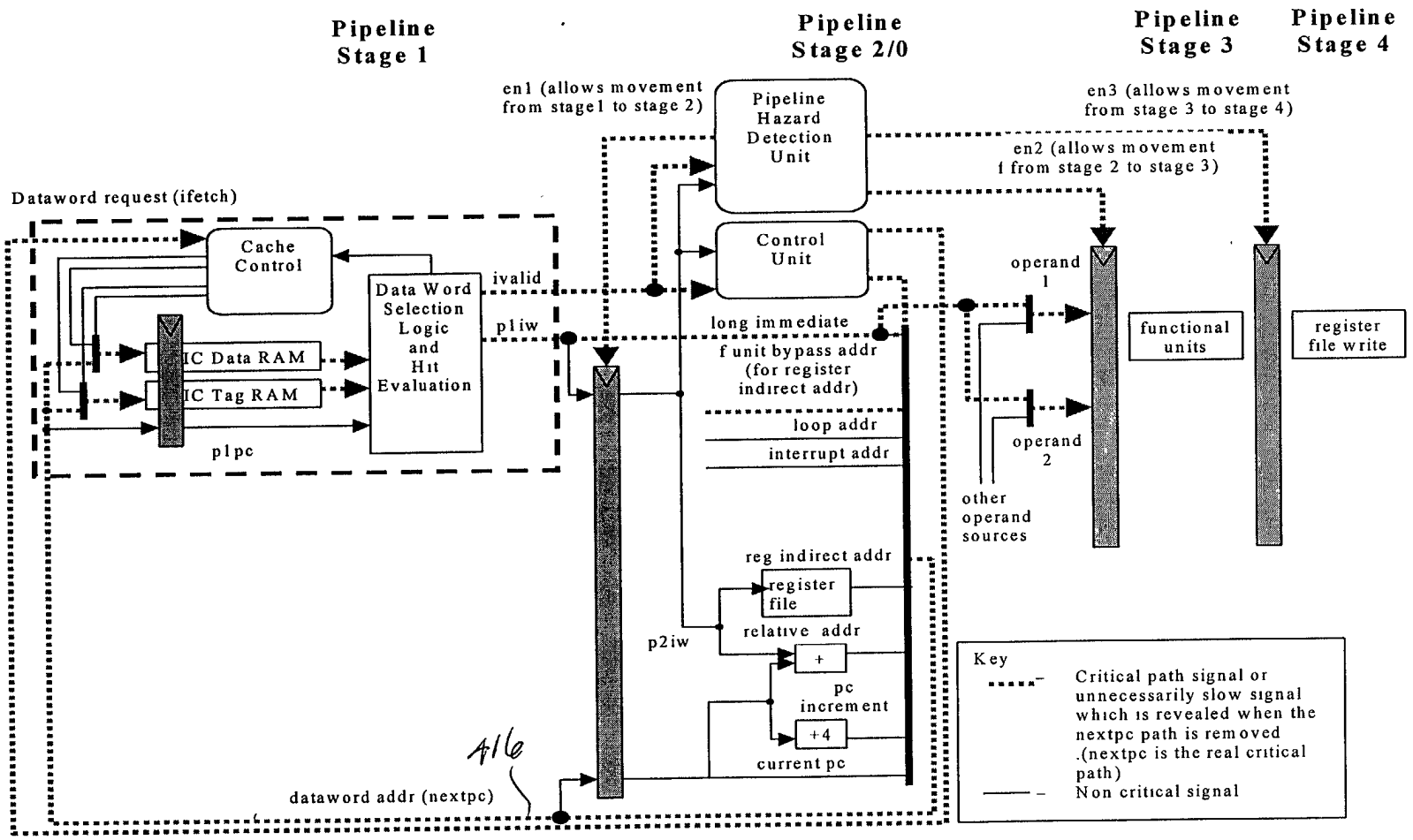
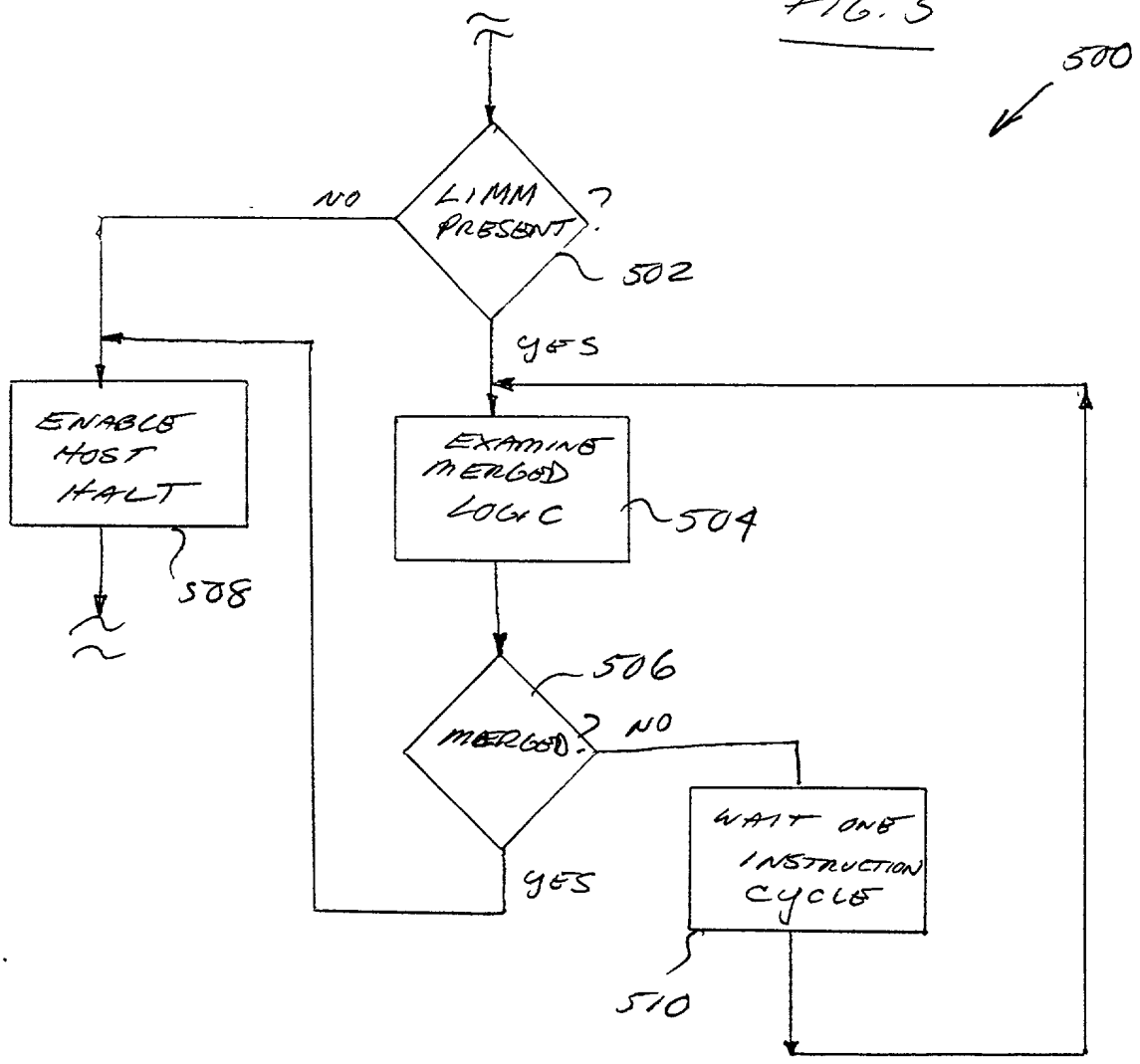

Pipeline Stage 1
Pipeline Stage 2/0
Pipeline Stage 3
Pipeline Stage 4
en1 (allows movement from stage1 to stage 2)
Pipeline Hazard Detection Unit
en3 (allows movement from stage 3 to stage 4)
en2 (allows movement from stage 2 to stage 3)
Dataword request (ifetch)
Cache Control
Control Unit
Data Word Selection Logic and Hit Evaluation
ivalid
p1iw
IC Data RAM
IC Tag RAM
p1pc
long immediate
f unit bypass addr (for register indirect addr)
loop addr
interrupt addr
operand 1
operand 2
other operand sources
functional units
register file write
reg indirect addr
register file
relative addr
+
pc increment
+4
current pc
p2iw
416
dataword addr (nextpc)
Key
Critical path signal or unnecessarily slow signal which is revealed when the nextpc path is removed .(nextpc is the real critical path)
Non critical signal

Fig. 4a

400

FIG. 5


500
LIMM PRESENT ?
NO
YES
502
ENABLE HOST HALT
508
EXAMINE MERGED LOGIC
504
MERGED ?
506
NO
YES
WAIT ONE INSTRUCTION CYCLE
510

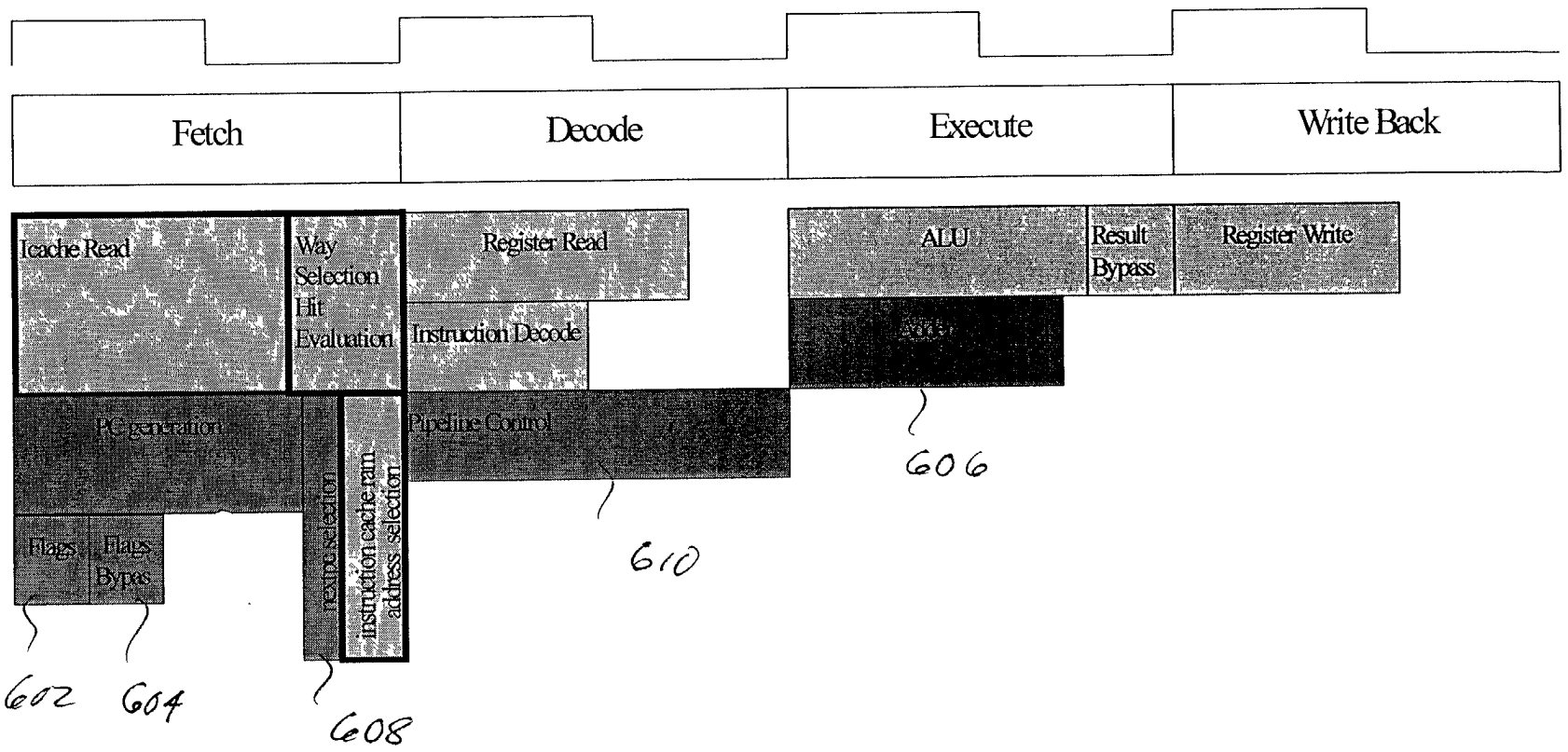

Fetch
Decode
Execute
Write Back
Icache Read
Way Selection Hit Evaluation
Register Read
Instruction Decode
ALU
Result Bypass
Register Write
PC generation
Pipeline Control
Flags
Flags Bypas
next pc selection
instruction cache ram address selection
602
604
608
610
606

Fig. 6

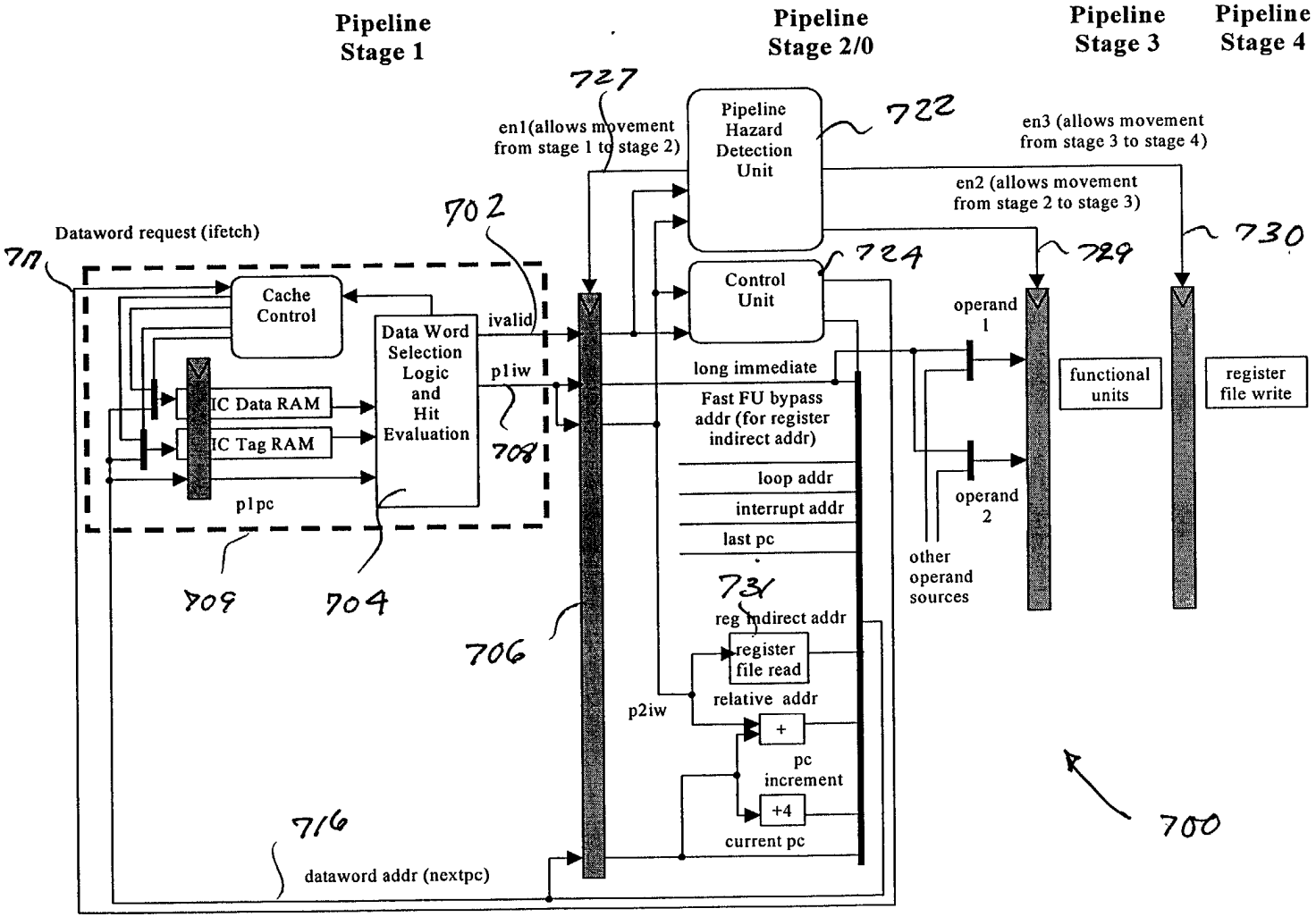

Pipeline Stage 1
Pipeline Stage 2/0
Pipeline Stage 3
Pipeline Stage 4
727
en1(allows movement from stage 1 to stage 2)
Pipeline Hazard Detection Unit
722
en3 (allows movement from stage 3 to stage 4)
en2 (allows movement from stage 2 to stage 3)
702
Dataword request (ifetch)
711
Cache Control
Data Word Selection Logic and Hit Evaluation
ivalid
p1iw
IC Data RAM
IC Tag RAM
p1pc
708
709
704
706
Control Unit
724
729
730
operand 1
long immediate
Fast FU bypass addr (for register indirect addr)
loop addr
interrupt addr
last pc
731
reg indirect addr
register file read
relative addr
p2iw
+
pc increment
+4
current pc
716
dataword addr (nextpc)
other operand sources
operand 2
functional units
register file write
700

Fig. 7

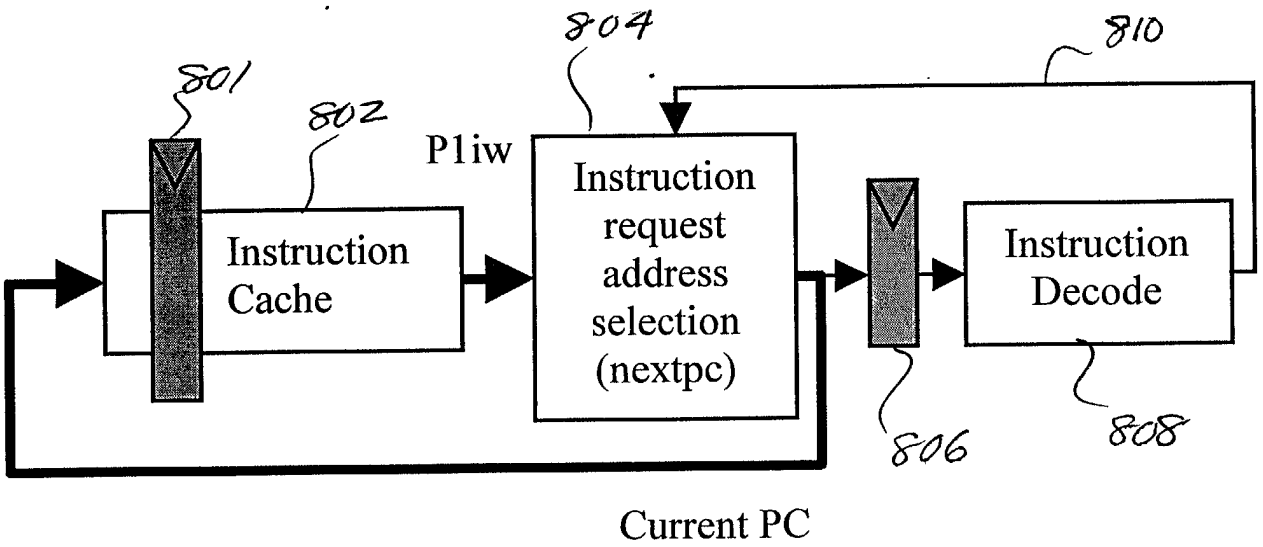

804
810
801
802
P1iw
Instruction request address selection (nextpc)
Instruction Cache
Instruction Decode
806
808
Current PC

Fig. 8

800

Fig. 8a

Reference Step	Instruction Fetch Address (Inst Addr: next_pc)	Pipeline Stage			
		Fetch (Inst Addr: current_pc)	Decode (Inst Addr: last_pc)	Execute	Write-back
820	$J.d_A$				
822	Brk_B	$J.d_A$			
824	$Target_C$	Brk_B	$J.d_A$		
826		$Target_C$	Brk_B	$J.d_A$	
828			Brk Restart		$J.d_A$
830			Brk Restart		
832	Add_B		Brk Restart		
834	$Target_C$	Add_B			
836	$Target2_C$	$Target_C$	Add_B		
838	$Target3_C$	$Target2_C$	$Target_C$	Add_B	

Fig. 8b

Reference Step	Instruction Fetch Address (Inst Addr: next_pc)	Pipeline Stage			
		Fetch (Inst Addr: currentpc)	Decode (Inst Addr: last_pc)	Execute	Write-back
840	Add_A				
842	Brk_B	Add_A			
844	Mov_C	Brk_B	Add_A		
846		Mov_C	Brk_B	Add_A	
848			Brk Restart		Add_A
850			Brk Restart		
852	Add_B		Brk Restart		
854	Mov_C	Add_B			
856	$Mov2_C$	Mov_C	Add_B		
858	$Mov3_C$	$Mov2_C$	Mov_C	Add_B	

Fig. 8c

Reference	Instruction Fetch Address (Inst Addr: next_pc)	Pipeline Stage			
		F (Inst Addr: currentpc)	D (Inst Addr: last_pc)	E	W
860	$J.d_A$				
862	Brk_B	$J.d_A$			
864	$Target_C$	Brk_B	$J.d_A$		
866		$Target_C$	Brk_B	$J.d_A$	
868			Brk Restart		
870			Brk Restart	$J.d_A$	
872	Add_B		Brk Restart	$J.d_A$	
874	$Target_C$	Add_B		$J.d_A$	
876	$Target2_C$	$Target_C$	Add_B	$J.d_A$	
878	$Target3_C$	$Target2_C$	$Target_C$	Add_B	$J.d_A$

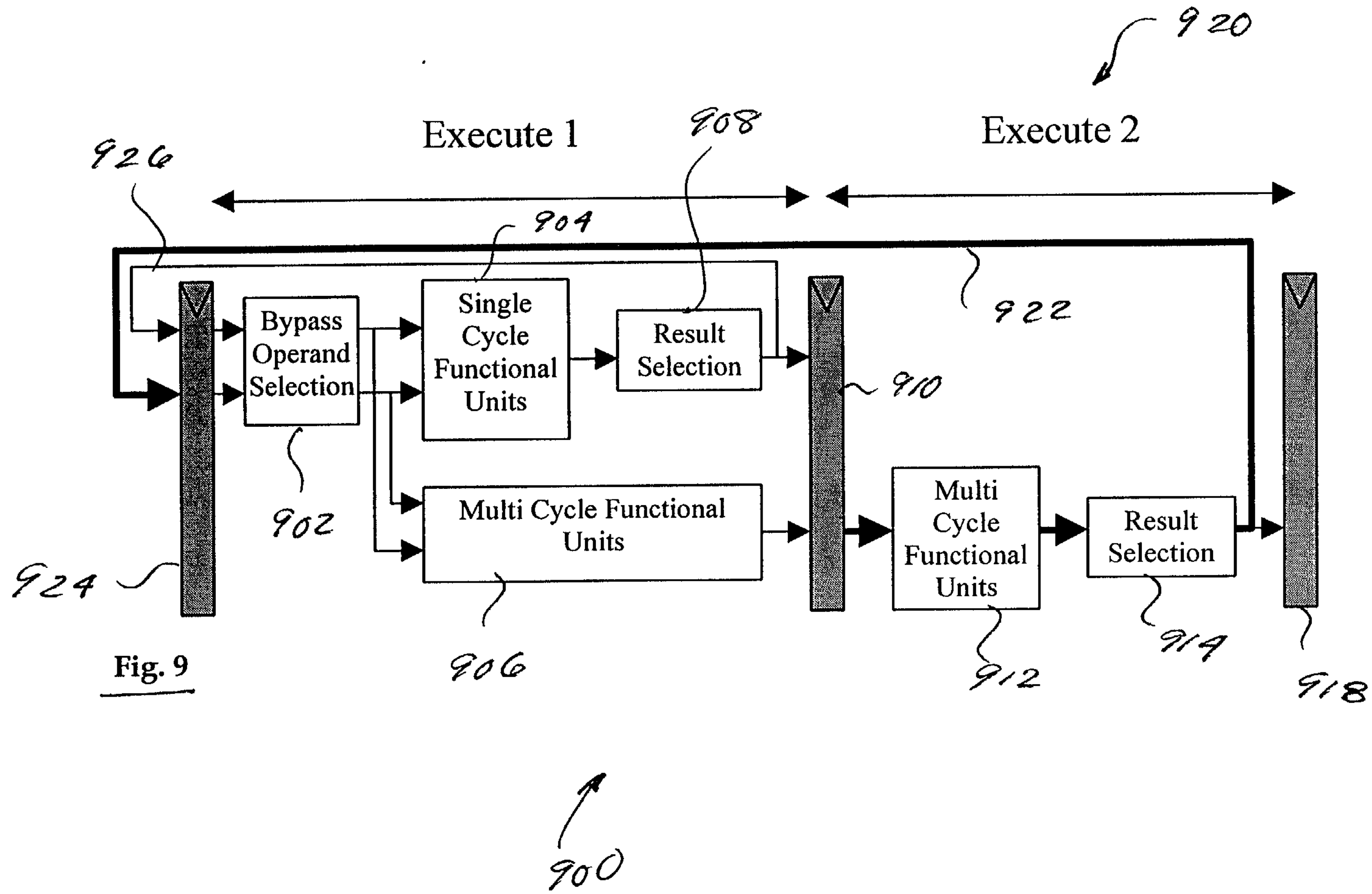
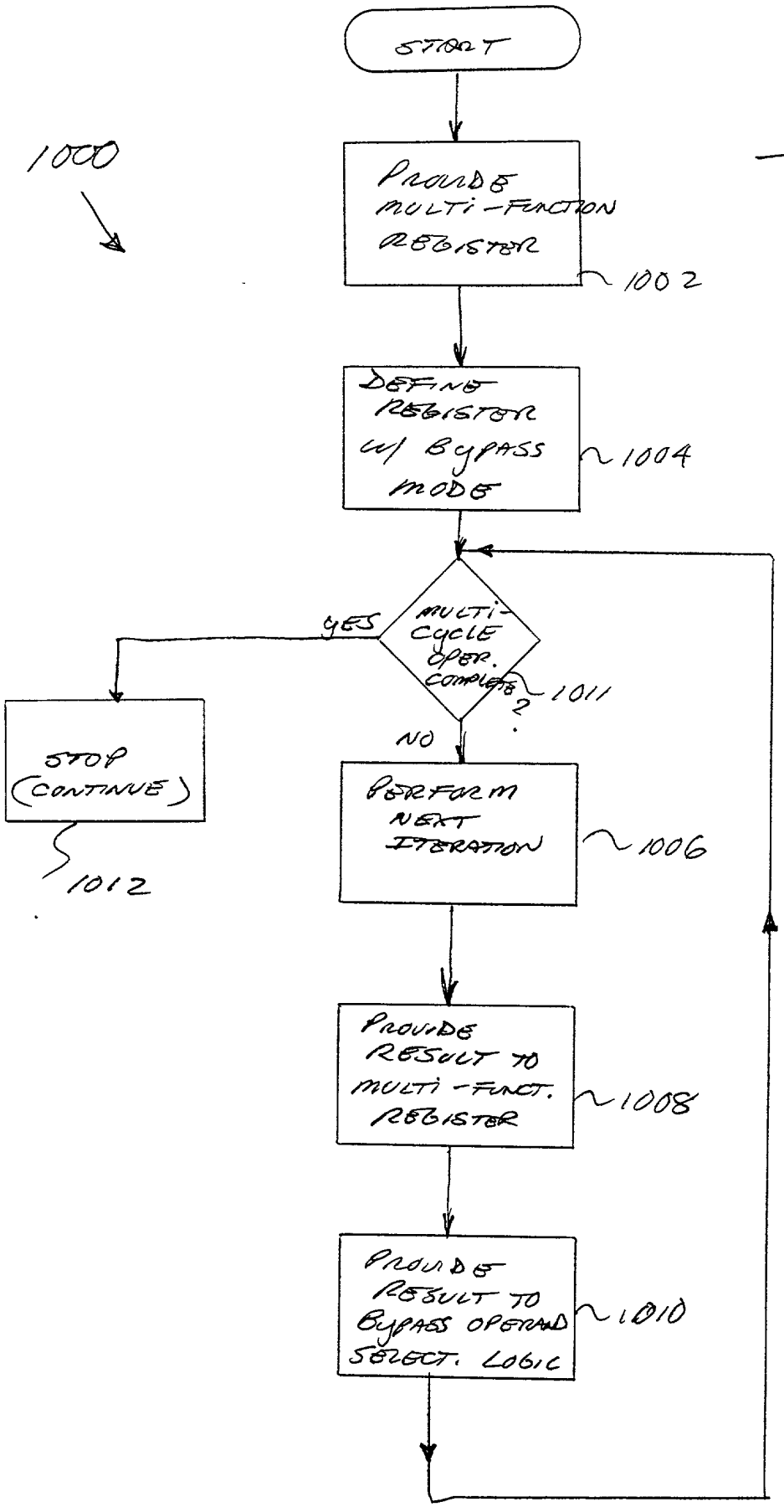

920
Execute 1
Execute 2
926
908
904
922
Bypass Operand Selection
Single Cycle Functional Units
Result Selection
910
Multi Cycle Functional Units
Multi Cycle Functional Units
Result Selection
902
924
906
912
914
918
900

Fig. 9

FIG. 10


1000
START
PROVIDE MULTI-FUNCTION REGISTER
1002
DEFINE REGISTER W/ BYPASS MODE
1004
MULTI-CYCLE OPER. COMPLETE?
1011
YES
NO
STOP (CONTINUE)
1012
PERFORM NEXT ITERATION
1006
PROVIDE RESULT TO MULTI-FUNCT. REGISTER
1008
PROVIDE RESULT TO BYPASS OPERAND SELECT. LOGIC
1010

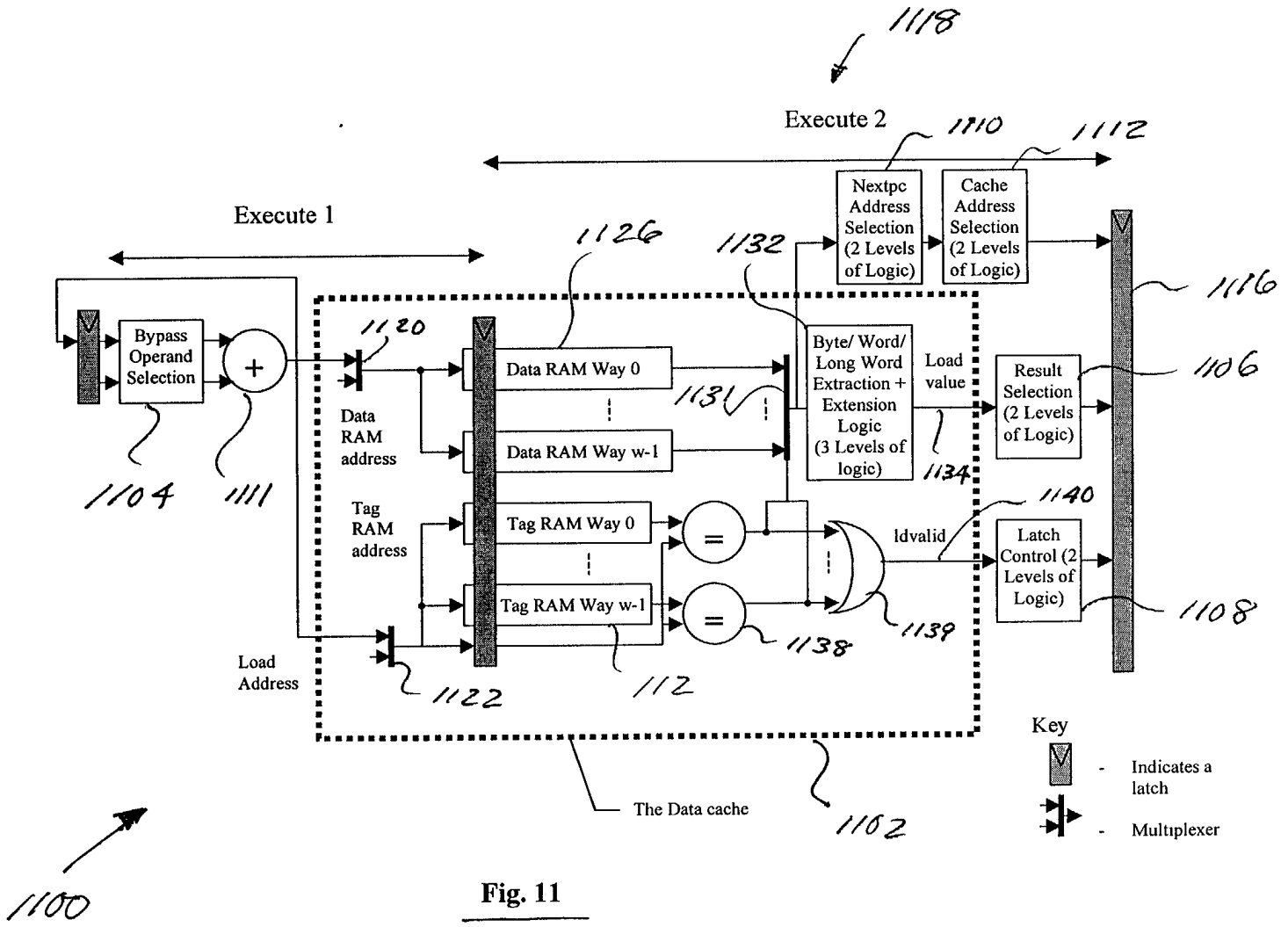

1118
Execute 2
1110
1112
Execute 1
1126
1132
Nextpc Address Selection (2 Levels of Logic)
Cache Address Selection (2 Levels of Logic)
1116
Bypass Operand Selection
+
1120
Data RAM Way 0
Data RAM Way w-1
Data RAM address
1131
Byte/ Word/ Long Word Extraction + Extension Logic (3 Levels of logic)
Load value
Result Selection (2 Levels of Logic)
1106
1104
1111
1134
1140
Tag RAM address
Tag RAM Way 0
Tag RAM Way w-1
=
=
Idvalid
Latch Control (2 Levels of Logic)
1139
1138
1108
Load Address
1122
112
The Data cache
1102
Key
- Indicates a latch
- Multiplexer
1100

Fig. 11

Fig. 11a

Reference	Pipeline Stage					
	0	F	D	E1	E2	WB
1174	Ld					
1176	Mov	Ld				
1178	Add	Mov	Ld			
1180		Add	Mov	Ld		
1182			Add	Mov	Ld	
1184				Add	Mov/Ld	
1186				Add	Ld	Mov
1188				Add		Ld
1190					Add	
1192						Add

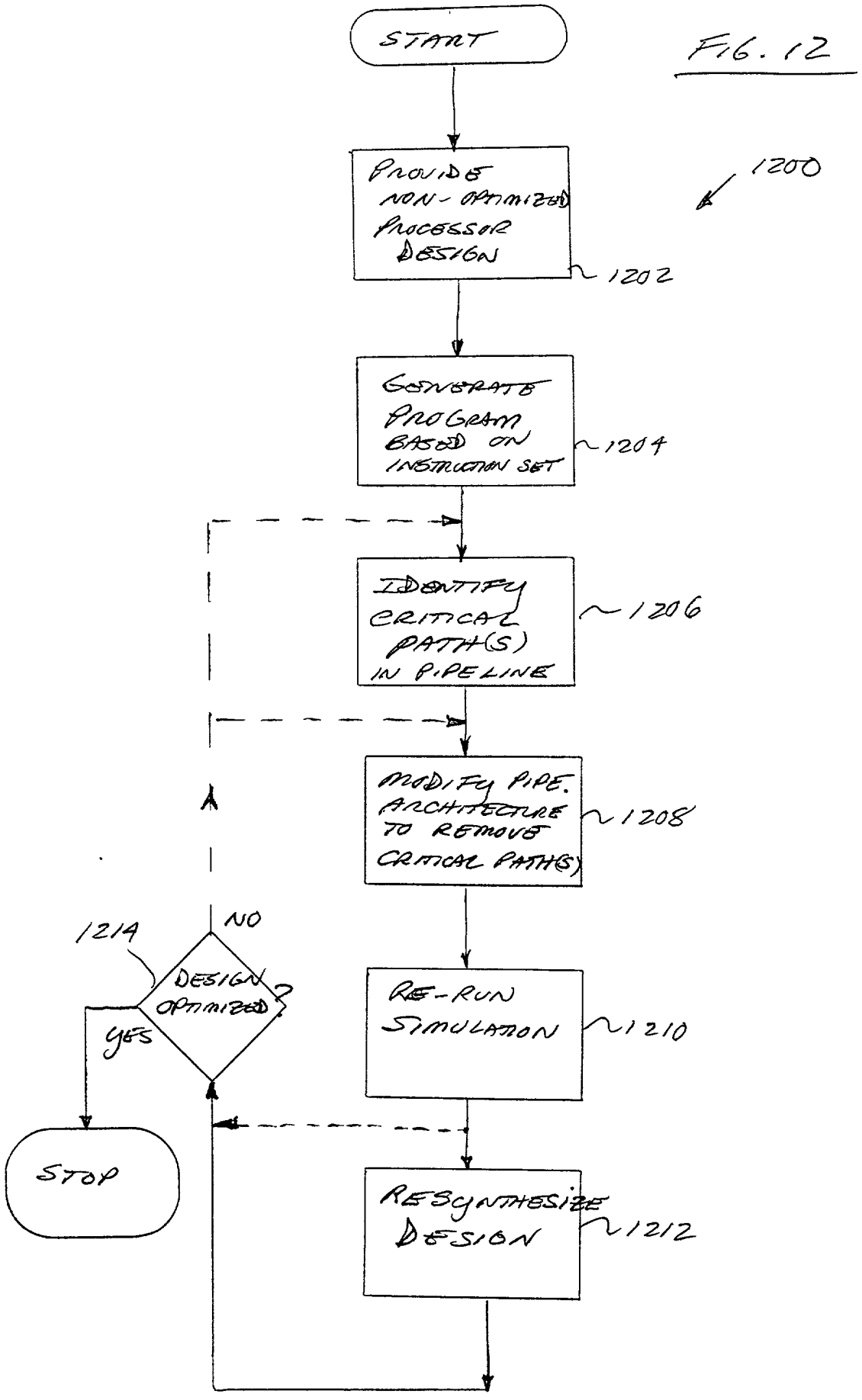

START
PROVIDE NON-OPTIMIZED PROCESSOR DESIGN
1202
1200
GENERATE PROGRAM BASED ON INSTRUCTION SET
1204
IDENTIFY CRITICAL PATH(S) IN PIPELINE
1206
MODIFY PIPE. ARCHITECTURE TO REMOVE CRITICAL PATH(S)
1208
RE-RUN SIMULATION
1210
RESYNTHESIZE DESIGN
1212
DESIGN OPTIMIZED?
1214
NO
YES
STOP

FIG. 12

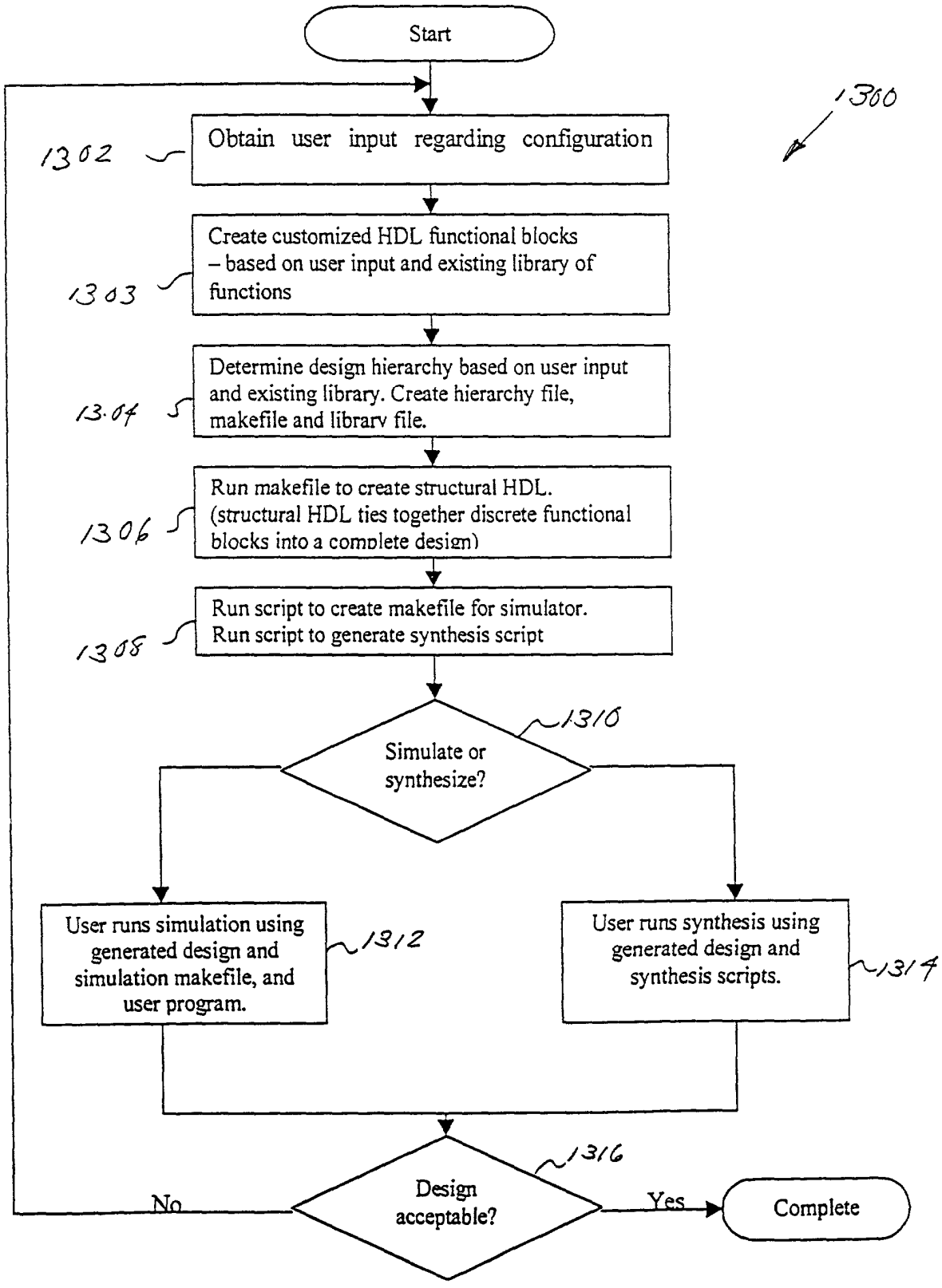

Start
1300
1302
Obtain user input regarding configuration
1303
Create customized HDL functional blocks – based on user input and existing library of functions
1304
Determine design hierarchy based on user input and existing library. Create hierarchy file, makefile and library file.
1306
Run makefile to create structural HDL. (structural HDL ties together discrete functional blocks into a complete design)
1308
Run script to create makefile for simulator. Run script to generate synthesis script
1310
Simulate or synthesize?
1312
User runs simulation using generated design and simulation makefile, and user program.
1314
User runs synthesis using generated design and synthesis scripts.
1316
Design acceptable?
No
Yes
Complete

FIG. 13

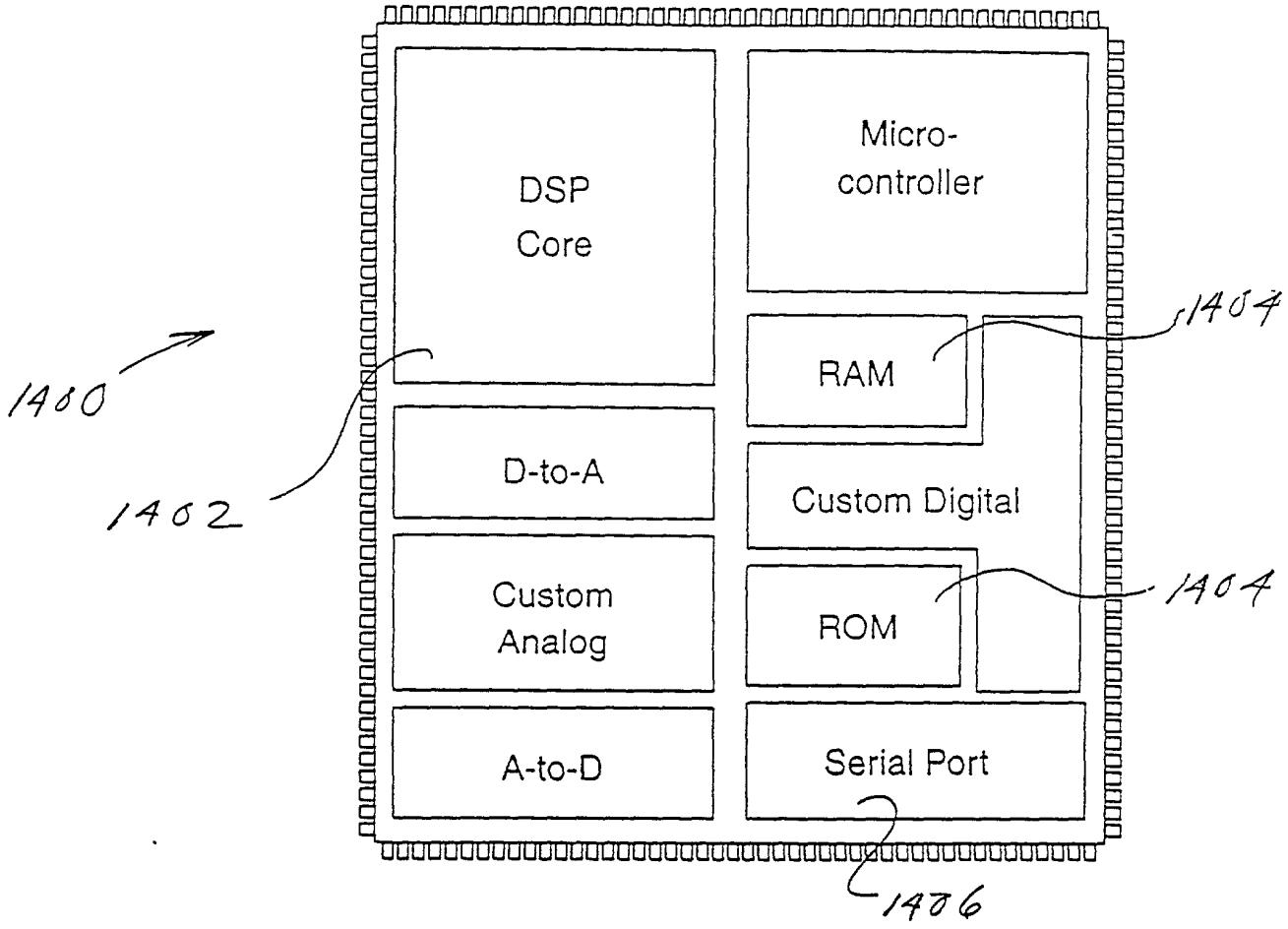

DSP
Core
Micro-
controller
RAM
D-to-A
Custom Digital
Custom
Analog
ROM
A-to-D
Serial Port
1400
1402
1404
1404
1406

FIG. 14

FIG. 15